Introduction

The present invention relates to gaming console architecture. The present invention seeks to provide an architecture which may be utilised on a wide variety of different platforms and which can reduce variation in games and allow them to use a wider variety of platform architectures.

Background of the Invention

The "combinations" of a game describe the mathematical structure of the game and define all possible games, including the winning patterns and the payouts associated with each. From the combinations, the game statistics are determined, including the theoretical return to the player.

As the terms are used in this document, "combinations" and "graphics and audio" may contain both data and code.

Currently a number of gaming architectures exist and are suitable for implementing games on a wide variety of platforms.

1. Standalone Electronic Gaming Machine (EGM). A standalone gaming machine contains all its functions within a secure environment. EGM's are commonly networked, primarily for data collection, bonusing and simple control.

2. Distributed gaming, such as Internet or In-room gaming (Hotel), separates the user interface (console) from the secure gaming server. High level communications reduces the need to send high bandwidth graphics data. A single server controls multiple consoles. Graphic and audio data necessary for game display are stored on the server and downloaded to the console as required for game display. To maintain security all functions that may effect game outcomes and accounting are performed on the server. A variation on this architecture is the distributed gaming accelerator, in which the gamble outcome decision logic is implemented at the console in a smartcard. Both server and console run combinations, generate/verify game outcomes and perform accounting.

3. TV allows the use of a television to play games. It separates the user interface (TV) from the rest of the gaming machine in a similar manner to the distributed gaming architecture, yet it could also be considered that a traditional EGM generates a display which is viewed via a remote TV screen. It requires more bandwidth than the distributed architecture as all picture information is generated at the server and sent in a low level, high bandwidth

form to the user interface. A low bandwidth communication channel is required to pass user input back to the server.

4. Hand held architecture. The secure functions of the EGM are implemented in a smartcard and all other functions in the unsecured console. The hand held architecture implements the secure functions within a smartcard. Limited storage capacity on the smartcard requires storage of graphics and combinations on the console for maximum flexibility. Variations on the storage location depend on the requirements (jurisdictional and otherwise). The smartcard may implement fixed combination(s) or download secured (encrypted or digitally signed) combinations from the console. When smartcard storage capacity is sufficiently large the smartcard may store graphics and audio for download to the console.

Table 1 shows the variety of platforms that may be implemented with these four fundamental gaming architectures.

Table 1 Gaming applications and architectures

	Architecture			
Platform	Traditional	Distributed	TV	Hand held
EGM	•	•		•
Hotel In-room		•		•
In-flight		•		•
On Ship	•	•	•	•
Internet		•		•
Cable TV		•	•	
Hand held				•

Split-Software Architecture

Referring to Figures 1 and 2, Split-software architecture has been previously proposed to allow games to be run independently of changes in the hardware platform. Further, the use of interpreted code allows games to be run on different microprocessors.

Split software architecture divides the gaming machine software into its two major elements. The game software contains all that software that is different between games, while the platform software contains the software that is common. In the traditional gaming machine, game software refers to the entire gaming machine software, while in the split software architecture, game software refers to that software that is different between "games". The exact meaning of game software is therefore

context dependent. Typically game code includes graphics and sound data and combinations, while platform code includes all hardware drivers, kernel, accounting, security, operator interface, communications, etc.

In principal the two separate parts of the game can be approved independently, so that:

- the platform EPROM is examined and approved only when it is changed and is tested with only a small subset of the possible games it will actually be used with. Hardware changes can be made to the platform with software changes limited to the platform. Games need not be re-approved.
- The game EPROM is examined and approved without considering the version of the platform it will be used with.

Platform upgrades are feasible as only a single EPROM(s) need be submitted (for each market) to allow all games to run.

The present invention seeks to provide a game architecture which may by utilised on a wide variety of different platforms, which they can reduce the amount of approvals required and simplify game creation.

Summary of the Invention

The present invention provides a gaming architecture including a game platform interface and a game program, the game program including a plurality of functional modules which interact via the platform interface. By providing such architecture in the game (software), the game can run on a variety of platform architectures without modifications of the game.

In one preferred embodiment, the game program includes a user interface module and a combinations module and communication of combinations to be displayed, are conveyed from the combinations module to the user interface module via the platform interface.

Brief Description of the Drawings

Embodiments of the invention will now be described, by way of example with reference to the accompanying drawings in which:

Figure 1 diagrammatically illustrates a Split-Software Architecture;

Figure 2 diagrammatically illustrates in greater detail the upper layers of the Split-software of Figure 1;

Figure 3 diagrammatically illustrates a variation in the upper layers of a Split-software architecture to provide a Multi-platform architecture in an EGM according to an embodiment of the present invention;

Figure 4 diagrammatically illustrates a variation in the arrangement of Figure 3 which is used in a Multi-platform Distributed Architecture;

Figure 5 diagrammatically illustrates a variation in the arrangement of Figure 3 which is used in a Multi-platform Standalone EGM;

Figure 6 diagrammatically illustrates Multi-platform Distributed Gaming System incorporating a number of different platforms; and

Figure 7 schematically illustrates the interconnection of a Game Server to the various components of a Multi-platform system.

Detailed Description of the Preferred Embodiments

In a multi-platform architecture, according to a preferred embodiment of the present invention, the game software is split into separate functions, such that the functions can be distributed to, and run on, each of the platforms for which it is required that the game support.

After removing the platform code, what remains of a traditional monolithic game is principally the combinations and the graphics/audio. By splitting the game software into separate combinations and graphics/audio code which always interact with each other via the platform, the game can be run on a wide range of platforms. As used herein, the term "user interface module" is the totality of the presentation of the game to the user, and usually comprises the graphics and sound.

A single software architecture is described below which is capable of supporting diverse platform requirements. In principal an approved game can be run on each of the platforms without modification, and approval of the game on one platform is sufficient for approval on all platforms.

The four traditional gaming architectures described differ in where each of these functions of the game is stored and executed. In a traditional EGM both functions are stored and executed within the EGM. In a distributed system the server stores the entire game, but executes only the combinations, while the console executes the downloaded graphics and audio. The distributed gaming accelerator system stores games on the secure server, executes combinations on both server and console, and graphics/audio on the user console. In the handheld system the entire game may be stored either on the console or smartcard or split between them depending on implementation, but combinations are executed on the smartcard and graphics/audio on the console. The TV system is identical in this respect to a traditional EGM, but the graphics are viewed on the remote TV.

Traditional game software is compatible only with the architecture and hence platform for which it was designed. It cannot run on multiple platforms, as the

components of the game are either monolithic (as in an EGM) or separated (as in a distributed system).

The two functions of the game, combinations and graphics/audio are separable, and the combinations may be secured (through encryption or digital signature or physically) to prevent tampering. The separation between the functions requires that the game API (application program interface) layer mediate communications between the functions.

When the game is run the game code is separated, as required by the platform architecture, into the appropriate functions and downloaded (where necessary) to the appropriate part of the platform. Figure 3 shows how the architecture is implemented in a traditional EGM. The combinations 10 and graphics and audio 12 are separated. They communicate with each other only through the API layer/platform interface 14.

Each of the separate game functions (eg. combinations, graphics/audio) may need to be secured to prevent tampering. If the function may be downloaded over an unsecured communication channel then the possibility of tampering exists. The consequences of tampering with combinations are particularly severe as it allows the payout of the game to be changed. Encrypting the data or creating a digital signature provides security. Encryption hides the data, however, a digital signature is generally quicker to authenticate.

In the distributed system the entire game is initially stored on the server. When the player requests a game, it is separated and graphics/audio are downloaded to the console and combinations are kept in the server. The same game in a traditional EGM is simply stored and executed unchanged. Figure 4 shows a distributed system generally indicated by 20, with one server 22 containing 'N' games 24 and controlling 'M' consoles 26. Clearly more than one server may be used.

Platform code is that software required to support a game (or part of a game), on a particular platform. It may perform the separation and distribution of game functions to those platforms for which it is required and provides communications where needed. Platform code exists for each platform within the gaming system and is in principal approved once for all games. A traditional EGM will have platform code for the EGM, while a distributed gaming system will have distinctly different platform code for the server and console(s). In practice platform code will typically contain code to recognise player inputs (push-buttons, handle, touch-screen, etc), drive player outputs (video, stepper reels, audio, etc) and drive machine accessories (printer, hopper, note-validator, security, etc). Depending on system implementation the system communications code

may also be considered part of the platform code, but has been drawn separately in the Figures to aid in understanding the systems.

Figure 5 shows an exemplary implementation of a standalone EGM using the multi-platform architecture and shows the separate game and platform approvals.

Figure 6 shows a distributed gaming system generally indicated by 50, comprising of a server 52, distributed EGM 54 , standalone EGM 56 and in-room gaming console 58. The architecture of distributed and in-room gaming EGM is essentially the same, with the main differences being in payment systems and physical design. Separate approvals are required for games (#1), server (#2), distributed EGM (#3), in-room EGM (#4) and standalone (#5) platform code. The standalone EGM may be monitored by and have games downloaded from the server.

In a casino, standalone EGM's are often connected to networks to allow monitoring and simple control. These networks can be extended to perform similar functions over the systems described, with, for example, distributed EGM's or the server itself being connected to these traditional networks. For compatibility the distributed EGM may emulate a traditional EGM. Alternately the network monitoring system may either be upgraded to understand the server or the server may emulate the appropriate number of standalone EGM's. Clearly both options may be implemented simultaneously.

In an extension to the architecture, the game may contain multiple graphic/audio and combination files, only one of which is used to play a particular game. One useful application is where various target platforms have different screen resolutions. Multiple graphics allow the best possible picture to be displayed. Where the target platform has only a very simple non-graphic interface one of the graphics files may cater for this. Different graphics may also allow the player to select their choice of "game". Different combinations cater for different player preferences, for example high win rates or large win values.

Partial replacements of the combinations and/or graphics/audio files allow the game to be partially modified, thus decreasing the storage requirements compared to storing complete copies of each possible games variation. For example, if a game is created that may be used with 50 different currencies a single main set of game graphics can be stored together with 49 different currency symbols. The total storage is far less than if 50 entire sets of game graphics were to be stored. Even worse, if 3 different symbols were to be selected, each from 50 possible, then the total number of variations is 125000 (50×50×50). Similarly audio and combinations may be partially replaced by equivalent data.

The multi-platform architecture allows embodiments incorporating other additional functions, which may be required if other aspects of a game are created or if new platforms are developed which require other functions of a game to run on one platform, but not another. In this way the multi-platform architecture can support a diverse range of platforms with a single game. The multi-platform architecture may be used in conjunction with interpreted code to achieve CPU independence on all platforms.

The architecture enables the creation of a generic "game server". The game server 62 stores games for execution and distribution to the various platforms, as shown in Figure 7. The game server may therefore be used to distribute games to traditional EGMs 64, Internet consoles 66, televisions 68, etc.

It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.